# Project 1 Report

The pr1 folder, which contains the source code, contains four python files:

1. **pr1_funcs.py:** This file contains all the functions necessary to compute the minimized SOP and POS expressions. Each function is systematic and computes individual components of the Quine-McCluskey algorithm for boolean logic minimization and Petrick's algorithm for solving cyclic PI tables. The functions are modularized in order to repeat calculations with minterms (for SOP) and maxterms (for POS) with ease.

    a. extract_minterms_dc (boolfunc): Given the boolean function as input, this function extracts the minterms and don't cares into lists and returns them.
    b. find_num_var(minterms, dc): Given the lists of minterms and don't cares, this function finds the number of variables by using the highest term included and returns this number.
    c. get_maxterms(minterms, dc, num_var): Adds any term that is not a minterm or a don't care to a list of maxterms and returns the list.
    d. function_output(minterms, dc, num_var): Creates and returns a list for the function output (1 for minterm, 0 for maxterm, and -1 for don't care).
    e. number_of_ones (num_var, func_output): Creates and returns a list indicating the number of 1 bits in each term.
    f. group_num_ones(num_ones, func_output, num_var): Groups terms with the same number of 1 bits. Returns a list of lists where each sub list contains the number and it's binary representation.
    g. findCubes(cubes, cube_num, prime_implicants, num_var): Given the cube number to calculate and the previously calculated cubes, this function generates the new cubes and checks off previously used cube. At the end, it adds any PIs that are not checked off to a PI list. All the cubes are stored in a single list, where cubes[0] are the 0 cubes and so on. Within the list of cubes are lists containing the grouped terms and the resulting binary expression where the bit with Hamming distance of 1 is replaced with an 'x'. It then appends the list of new cubes to the inputted list of cubes, which it returns along with the list of prime implicants.
    h. generate_coverage_table(prime_implicants, minterms): This function generates a table with the rows being the PIs and the columns being the minterms and assigns a 0 or 1 for each spot depending on whether the minterm appears in the PI.
    i. find_EPIs_covered_minterms(coverage_table, minterms, prime_implicants): From the coverage table, if a minterm is covered by only 1 PI, the corresponding PI is added to a list of essential PIs and all the minterms covered by that PI are added to a list of covered minterms. Both these lists are returned.

j. remove_EPIs_covered_minterms(prime_implicants, essential_PIs, dc, covered_minterms): This function makes a reduced coverage table by removing the rows corresponding to the essential PIs and the columns corresponding to the covered minterms and returns it. It also assigns letters to the unessential PIs.
k. get_uncovered_minterms(minterms, covered_minterms): This function looks through the list of all minterms and returns a new list that only contains the minterms that are not in the list of covered minterms.
l. petricks_terms(uncovered_minterms, reduced_table): From the reduced table and list of uncovered minterms, this function groups the assigned letters corresponding to the unessential PIs for each minterm into lists, and a list of all these groupings is returned.
m. initial_combine(petricks): Uses the boolean law (X + Y)(X + Z) = (X + YZ) to combine groups of terms from the petricks list. Returns the simplified list.
n. multiply_terms(a,b): Given any two terms, this function multiplies the two terms, removes any redundant terms using the boolean logic rule XX = X, and returns the multiplied result, where each term is an element in a list.
o. distribute_all_terms(petricks): From the simplified petricks list, this function repeatedly calls the above multiply terms function to distribute entirely and return a single combination of terms.
p. logic_combination(distributive): In the distributed list of terms, this function uses the boolean law X + XY = X to remove redundant terms and returns the new list.
q. min_PI_from_Petricks(combine_terms, reduced_table, essential_PIs): From the list of combined terms, this function finds the combined term with the lowest cost based on the corresponding PI that the letter represents and returns that term.
r. PI_to_SOP(PI_list): Given the list of PIs, this function assigns letters A,B,C,... for the variables, A/A' depending on if the bit is a 0 or 1, and groups them in SOP format.
s. PI_to_POS(PI_list): Similar to the above function, it assigns variable letters and groups the PIs in POS format.

2. **Pr1_input.py:** To provide an intuitive and aesthetic way of providing input, I used the tkinter package to generate a user interface. Tkinter provides frames, labels, buttons, textboxes, and many other components needed to build a UI. Upon loading the program, the root frame is shown with two buttons, corresponding to manual or file input. If the file option is chosen, the tkinter file finder function is called to display a file search box and return the path of the selected file, which is then opened and read. Files can contain multiple boolean functions following the format m(...)+d(...). Each of these functions is read into a list which can then be called in the main file. If the manual option is chosen, a second frame is displayed with two input boxes for the minterms and don't cares. The

values, which should be entered with a single space between them, are loaded into a string and sent to main.

3. **pr1_output.py:** This file gets the list of outputted functions from the main file and displays them on a tkinter scrollable frame. Each outputted function is added as a new label to the frame. The frame also indicates that step by step calculations are shown on the terminal screen.

4. **pr1_main.py:** This file imports the above three files and gets the input function(s) from pr1_input.py. For each inputted logic function, it calls the functions from pr1_funcs.py in the order they appear. First, functions a-f are called to set the basis for the Quine-McCluskey algorithm. Then functions g-k are called using the minterms to find essential PIs and the minterms covered by them using a coverage table. Functions l-q, which implement Petrick's method, are only called if the essential PIs don't cover every minterm. After finding the PI with the lowest cost from Petrick's method, this term is added to the list of essential PIs, and all the PIs are converted into SOP format. Functions g-q are repeated using the list of maxterms to similarly get the POS result. At each step, the values returned by the functions are stored and printed to the terminal for debugging purposes and to make the steps available for users who wish to follow along.